Переменная

Определение

Программа, которая всегда считает результат вычисления одного и того же выражения, довольно скучная и бессмысленная. Полезная программа должна оперировать с различными данными без внесения изменений в код.

Для хранения информации в программировании используются **переменные**. **Переменную** можно рассматривать как **ящик**. Однажды сделав такой ящик, мы можем класть в него разные вещи.

Под каждый тип информации – нужен ящик соответствующего типа: вы ведь не будете складывать деньги, спички, бензин и шоколад в один и тот же ящик. Таким образом, у каждой переменной есть **тип данных**, который надо указать при ее создании. Также у переменной есть **имя**, по которому мы будем к ней обращаться.

Работа с переменной

Для **создания** или **объявления** переменной нужно воспользоваться следующей схемой:

```language-kotlin
(ключевое слово [var или val]) название_переменной: тип_данных
```

Тип данных, отвечающий за хранение целых чисел называется `Int`. Следовательно, чтобы создать переменную, которая будет хранить **целые числа**, нужно написать:

```language-kotlin
var a: Int
```

Это мы **объявили переменную** с названием `a` и указали, что там будут храниться **целые числа**. Теперь в созданную переменную можем записывать только **целые числа.**

Чтобы записать в переменную целое число, нужно воспользоваться следующим правилом:

куда = что

где

- **куда**: в какую переменную записать данные
- `=`: **оператор присвоения**
- **что**: какие данные записать

Воспользуемся данным правилом. Например, запишем в ранее созданную переменную `a` число 77:

```kotlin
a = 7
```

Операции производятся **справа налево** – взять число 77 и записать в переменную `a`.

Эти два шага можно объединить, то есть можно сразу объявить переменную и записать в нее значение (**инициализировать**), иначе говоря – присвоить начальное значение:

```kotlin
var a: Int = 7
```

Мы объявили переменную `a` и сразу записали значение 77.

Чаще всего так и делают: **сразу объявляют переменную и присваивают начальное значение!**
И в таком случае можно немного сократить код:

```kotlin
var a = 7 // Тип данных определяется автоматически
```

В примере выше мы создаем переменную не указывая тип данных. Мы можем воспользоваться такой схемой если значение в переменную записывается сразу. В таком случае программа сама понимает что тип данных должен быть, например, `Int` (как в этом случае).

В этом курсе тип данных часто будет указываться **явно,** несмотря на то что в большинстве случаев тип данных не указывают. Сделано это было для лучшего понимания языка программирования.

Чтобы узнать **содержимое переменной**, нужно обратиться к ней по **имени**. Например:

```kotlin
var b = a * 5
```

В переменную `b` запишется значение 3535, так как вместо переменной `a` подставится ее **значение**, то есть 77. Напомню, что сначала выполняется выражение справа от равно (`a * 5`), а потом результат вычисления записывается в новую переменную `b`.

Также мы можем поменять значение уже существующей переменной `a`:

```kotlin
a = a + 8
```

Так как действия выполняются **справа налево от знака** `=`, следовательно, мы берем значение переменной `a`, которое равно 77, к нему добавляем 88 и снова записываем в переменную `a`. Таким образом, значение переменной увеличили на 88.

Переменную создают **один** раз, указав ключевое слово (`var` или `val`), тип данных, название и начальное значение. При изменении значения переменной, нужно указать только название. Тип данных и ключевое слово указывать больше не нужно.

Для лучшего понимания разберем пример с ошибкой:

```kotlin
var a: Int = 6 // создали переменную a

var b: Int = a * 8 // 48

var a: Int = b - 8 // ⛔ Ошибка. Переменная a уже существует.
var a = b - 8      // ⛔ Ошибка. Переменная a уже существует.
a: Int = b - 8     // ⛔⛔ Ошибка. Так вообще нельзя записывать.

var c: Int = b + a

println(c)
```

А вот исправленная программа:

```kotlin
var a: Int = 6

var b: Int = a * 8  // 48

a = b - 8           // 40. Заметьте, что тип данных мы не написали

var c = b + a       // 88

println(c)          // вывод 88
```

`val` и `var` - Отличия ключевых слов

Итак, мы уже знаем что при создании переменной нам нужно использовать ключевое слово - val или var. Теперь давайте разберемся, в чем их отличие.

Пока что мы использовали только `var`, потому что если вы знакомы с каким-нибудь си подобным языком (C#, Java, C++), то поведение такой переменной вам будет более понятно.

С помощью ключевого слова `**val**` создается переменная, которую нельзя в дальнейшем перезаписать. Если вы знаете что значение в переменной не изменится, лучше использовать его. Так меньше вероятность что вы выстрелите себе в ногу (допустите ошибку).

С помощью ключевого слова `**var**` создается переменная, в которую можно записывать данные того типа, который был указан при инициализации, и столько раз, сколько потребуется.

Пример с ошибкой:

```kotlin
val a: Int = 6      // создали переменную с неизменяемым значением
var b: Int = a * 8  // 48

a = b - 8           // ⛔ Ошибка. a - неизменяемая переменная
b = b - 8           // ✅

var c = a + b.      // ⚠ c в данной программе не изменяется. Желательно объявлять через val
println(c)
```

Исправленная программа

```kotlin
var a: Int = 6     // ✅
```

```
var b: Int = a * 8 // 48

a = b - 8          // ✅ 40
b = b - 8          // ✅ 40

val c = a + b      // ✅ 80
println(c)
```

Типы данных

Итак, мы уже посмотрели и поработали с целочисленным типом данных `Int`. Но существуют еще и другие типы данных. Давайте рассмотрим основные часто используемые типы данных:

- **Int** – целое число от `-2147483648` до `2147483647`;
- **Long** – целое число от `-9 223 372 036 854 775 808` до `9 223 372 036 854 775 807`;
- **String** – строка;
- **Double** – число с дробной частью (вещественные числа);
- **Char** – символ;
- **Boolean** – специальный тип, принимает только два значения: `true` или `false`.

Рассмотрим пример:

- Создадим переменную строкового типа `name` и запишем в нее строку "**Олег**". Напомню, что строку нужно обрамлять двойными кавычками:

```
val name: String = "Олег"                                    language-kotlin
```

- Создадим еще одну переменную целочисленного типа `age` и запишем в нее значение 2525:

```
val name: String = "Олег"                                    language-kotlin
val age: Int = 25
```

- Создадим еще одну переменную `weight` вещественного типа и запишем в нее значение 80.5:

```
val name: String = "Олег"                                    language-kotlin
val age: Int = 25
val weight: Double = 80.5
```

Стоит отметить, что при инициализации вещественных чисел дробная часть отделяется от целой **точкой**.

- Теперь мы можем обращаться к переменным по имени и вместо них подставятся их значения:

```language-kotlin
val name: String = "Олег"
val age = 25
val weight = 80.5
val stringToShow: String = name + ", возраст " + age + ", вес " + weight

println(stringToShow)
```

Запятые я поставил просто для красоты отображения. Они стоят внутри строки. Все что стоит внутри строки выводится в таком же виде. То есть программа выведет:

```language-undefined
Олег, возраст 25, вес 80.5
```

Рассмотрим подробнее. В переменную целого типа можно записывать **только** целое число:

```language-kotlin
var a: Int = 10
```

Нельзя записать в нее строку или любой другой тип данных:

```language-kotlin
var a: Int = "10"       // ❌ ошибка!
```

То же самое, например, с переменной строкового типа. В нее можно записать только строку:

```language-kotlin
val s: String = "test"  // ✅ верно

val t: String = 10      // ❌ ошибка!
```

Например, для строк знак плюс (`+`) означает **склеивание** между собой, а для целых и вещественных чисел плюс **складывает** их математически:

```kotlin
val a: Int = 5
val b: Int = 10
println(a + b) // 15

val s1: String = "5"
val s2: String = "10"
println(s1 + s2) // 510
```

Имя переменной

1. В имени переменной используйте только латинские буквы `a-z`, `A-Z`, цифры и символ нижнего подчеркивания (`_`);
2. Имя переменной не может начинаться с цифры;
3. Имя переменной по возможности должно отражать её назначение. Это нужно только нам, программистам, чтобы наш код читался. Программе все равно, какое название будет у переменной.

`Kotlin` — **регистрочувствительный** язык. Переменная `name` и `Name` — две совершенно разные переменные. Принято название переменных начинать с маленькой буквы.

Есть еще одно правило именования переменной. Имя переменной пишется в нижнем регистре. Если имя состоит из нескольких слов, то первая буква каждого слова, кроме первого, пишется в верхнем регистре. Это правило **не строгое.** Если писать как-то по другому, то программа все равно запустится (если соблюдать 1-3 пункты), но рекомендуется соблюдать это правило, чтобы поддерживать общий стиль кода для языка `Kotlin`:

```kotlin
// Примеры хороших имен переменных ✅
val firstName = "..."

val dayOfWeek = 7

val currentYear = 2023

// Примеры плохих имен переменных ⚠
val FirstName = "..."

val day_of_week = 7

val currentYEAR = 2023
```

Примечания:

1. Переменные можно вводить в любой момент (не только в самом начале программы).
2. Названия переменных должны быть уникальными. Если у двух переменных будут одинаковые имена, то непонятно будет, какое значение вернется при обращении к переменной.
3. Значение перезаписывается. Новое значение переменной вытесняет старое. Важно понимать, чему равно значение переменной в каждый момент времени:

```language-kotlin
var a: Int = 6      // здесь a равен 6

val b: Int = a * 8

a = b - 8           // 40. Здесь уже a равен 40

val c: Int = b + a

println(c)
```

4. Можно сразу записывать в объявляемую переменную формулу:

```language-kotlin
val a: Int = 22 * 4 + 1      // 89

val b: Int = a - 15 * 3      // 44
```

5. Напомню, что мы все написанные программы пишем внутри `main`.